ALTAS
MANUAL DE USUARIO
SERVICIO NOTE

Versión 1.0

Unidad de Arquitectura del Software

Agencia de Informática y Comunicaciones de la Comunidad de Madrid

Hoja de Control

Título	Manual de Usuario Servicio de NOTE		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	Unidad de Arquitectura del Software		
Versión	1.0	**Fecha Versión**	09/06/2014

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Unidad de Arquitectura del Software	09/06/2014

Índice

1. INTRODUCCIÓN 4

1.1. AUDIENCIA OBJETIVO 4
1.2. CONOCIMIENTOS PREVIOS 4

2. DESCRIPCIÓN 5

3. INSTALACIÓN Y CONFIGURACIÓN 6

3.1. INSTALACIÓN 6
3.1.1. Paso 1: Añadir la dependencia al módulo de Administración Electrónica 6
3.1.2. Paso 2: Incluir la ruta de definición del servicio 'NoteService' 7
3.1.3. Paso 4: Inyectar el Servicio en las clases que lo requieran 7

4. USO DEL SERVICIO NOTESERVICE 8

4.1. MÉTODO ENVIARNOTIFICACION 8
4.2. MÉTODO ENVIARNOTIFICACIONADJUNTOUNICO 8
4.3. MÉTODO ENVIARREMESA 9
4.4. MÉTODO ENVIARREMESAADJUNTOUNICO 11
4.5. MÉTODO OBTENERNOTIFICACIONESPENDIENTES 12
4.6. MÉTODO OBTENERNOTIFICACIONESPENDIENTESCONASUNTO 13
4.7. MÉTODO ELIMINARNOTIFICACIONESPENDIENTES 14
4.8. MÉTODO OBTENERDUE 15
4.9. MÉTODO EXISTEABONADO 15

5. ENLACES RELACIONADOS 16

Contenido

1. INTRODUCCIÓN

Este documento contiene instrucciones y ejemplos de uso de la librería de servicios de NOTE. Para más información sobre las capacidades funcionales de este servicio, debe consultarse la documentación original del proyecto.

La librería se compone del siguiente servicio:

- **NoteService** – Servicios de administración electrónica para interactuar con la aplicación que gestiona notificaciones telemáticas

1.1. Audiencia objetivo

El lector objetivo de este documento es toda aquella persona que esté desarrollando una solución en la que se necesite usar servicios de NOTE en sus aplicación ATLAS.

1.2. Conocimientos Previos

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- Lenguaje Java
- Spring Framework

2. DESCRIPCIÓN

El servicio de Spring "**NoteService**" que incluye el módulo de administración electrónica de Atlas, provee las siguientes funcionalidades, que se explican en los siguientes apartados:

- enviarNotificacion
- enviarNotificacionAdjuntoUnico
- enviarRemesa
- enviarRemesaAdjuntoUnico
- obtenerNotificacionesPendientes
- obtenerNotificacionesPendientesConAsunto
- eliminarNotificacionesPendientes
- obtenerDUE
- existeAbonado

3. INSTALACIÓN Y CONFIGURACIÓN

3.1. INSTALACIÓN

El Servicio '***NoteService*** está incluido en la librería ***atlasfrm-eadm-lib***. Para poder utilizarlo, bastará con incluir la dependencia de este módulo en el fichero *pom.xml* del proyecto, añadir la ruta correspondiente al fichero *xml* de Spring en la variable '*configLocations'*, y definir en el contexto de Spring la configuración del servicio para la aplicación., según se describe en los siguientes apartados:

3.1.1. Paso 1: Añadir la dependencia al módulo de Administración Electrónica

Atención
Este paso no es necesario si el proyecto se ha generado a partir del arquetipo *web, servicioweb* o *documentumweb* de Atlas, ya que dicha librería ya viene incluida por defecto en dichos arquetipos.

Para añadir la dependencia al módulo de seguridad dentro de nuestro proyecto *maven* hay que añadir una entrada "**dependency**" en la sección "**dependencies**" del fichero "**pom.xml**" del proyecto, como se puede ver en el siguiente ejemplo:

pom.xml

```
<dependencies>
	[...]
	<dependency>
	    <groupId>atlasfrm</groupId>
	    <artifactId>atlasfrm-eadm-lib</artifactId>
	    <version>${atlasfrm-eadm-lib.version}</version>
	</dependency>
	[...]
</dependencies>
```

3.1.2. Paso 2: Incluir la ruta de definición del servicio 'NoteService'

La librería de administración electrónica ya contiene una definición del servicio '**NoteService'** dentro del contexto de Spring. Bastará con incluir la siguiente ruta en una sentencia <import> dentro de un fichero de configuración de Spring (por ejemplo, en el fichero applicationContext-general.xml):

applicationContext-general.xml

```
<import resource="classpath:/conf/applicationContext-eadm-lib.xml"/>
```

3.1.3. Paso 4: Inyectar el Servicio en las clases que lo requieran

Para inyectar el Servicio previamente definido, se pasa como "**property**" al bean que lo vaya a usar, que debe tener el método "**set**" correspondiente. Para más información sobre cómo inyectar referencias, se puede consultar la documentación de Spring.

applicationContext-services.xml

```
<bean id="miService" class="xxxx.services.MiServiceImpl"
      p:eregAltaService-ref="noteService"/>
```

xxxx.services.MiServiceImpl.java

```
public class MiServiceImpl implements MiService {

    private NoteService noteService;

    public void setNoteService(NoteService noteService) {
        this.noteService = noteService;
    }
    ...
}
```

4. USO DEL SERVICIO NoteService

Una vez configurado el entorno para el uso del servicio, puede procederse a su utilización.

4.1. Método enviarNotificacion

NoteService.java

```
        /**
     * Método para enviar una notificación
     *
     * @param datosEntrada de tipo {@link NoteEnviarNotificacionIN}
     * @throws ServiceException
     */
    public void enviarNotificacion(NoteEnviarNotificacionIN datosEntrada) throws
ServiceException;
```

Parámetro de entrada

NoteEnviarNotificacionIN.java

```
/**
* Objeto con los datos de entrada para el método enviarNotificacion de
* {@link
atlas.eadm.services.PtfrService#enviarNotificacion(NoteEnviarNotificacionIN)}
* codigoApp Código de la aplicación
* codigoOrigen Referencia
* codigoOrigenInt Referencia interna
* cifrada Cifrado del cuerpo  del mensaje, Valores posibles:  true/false o 1/0
* refAportacion Nº Referencia de la aportación
* uniAdmin Descripción unidad Administrativa
* servicio Descripción servicio
* asunto Descripción asunto
* cuerpo Cuerpo de la notificación
* codigoAbonado DUE del interesado
* lista de listaNombresAdjunto (String) Nombre de los adjuntos
* lista de listaDatosAdjuntos {InputStream} Datos de los adjuntos en base64
*/
public class NoteEnviarNotificacionIN {
```

4.2. Método enviarNotificacionAdjuntoUnico

NoteService.java

```
    /**
     * Método para enviar una notificación con un solo adjunto
     *
     * @param datosEntrada de tipo {@link NoteEnviarNotificacionAdjuntosUnicoIN}
     * @throws ServiceException
     */
    public void enviarNotificacionAdjuntoUnico(NoteEnviarNotificacionAdjuntosUnicoIN
datosEntrada)
            throws ServiceException;
```

Parámetro de entrada

NoteEnviarNotificacionIN.java

```
/**
* Objeto con los datos de entrada para el método enviarNotificacionAdjuntoUnico de
* {@link
atlas.eadm.services.NoteService#enviarNotificacionAdjuntoUnico(PTFRAtlaSolicitudIN)}
* codigoApp Código de la aplicación
* codigoOrigen Referencia
* codigoOrigenInt Referencia interna
* cifrada Cifrado del cuerpo  del mensaje, Valores posibles:  true/false o 1/0
* refAportacion Nº Referencia de la aportación
* uniAdmin Descripción unidad Administrativa
* servicio Descripción servicio
* asunto Descripción asunto
* cuerpo Cuerpo de la notificación
* codigoAbonado DUE del interesado
* nombreAdjunto
* fichero con dato Adjunto {InputStream}
*/
public class NoteEnviarNotificacionAdjuntosUnicoIN
```

4.3. Método enviarRemesa

NoteService.java

```
    /**
     * Método que envía una remesa de notificaciones
     *
     * @param datosEntrada de tipo {@link NoteEnviarRemesaIN}
     * @throws ServiceException
     */
    public void enviarRemesa(NoteEnviarRemesaIN datosEntrada) throws
ServiceException;
```

Parámetro de entrada

NoteEnviarNotificacionIN.java

```
/**
* Objeto con los datos de entrada para el método enviarNotificacionAdjuntoUnico de
* {@link
atlas.eadm.services.NoteService#enviarNotificacionAdjuntoUnico(PTFRAtlaSolicitudIN)}
* codigoApp Código de la aplicación
* codigoOrigen Referencia
* codigoOrigenInt Referencia interna
* cifrada Cifrado del cuerpo  del mensaje, Valores posibles:  true/false o 1/0
* refAportacion Nº Referencia de la aportación
* uniAdmin Descripción unidad Administrativa
* servicio Descripción servicio
* asunto Descripción asunto
* cuerpo Cuerpo de la notificación
* codigoAbonado DUE del interesado
* nombreAdjunto Nombre del adjunto
* fichero con dato Adjunto {InputStream}
*/
public class NoteEnviarNotificacionAdjuntosUnicoIN
```

NoteNotificacionRemesaListaAdjuntos.java

```
/**
* Objeto con los datos necesarios para una notificación con varios adjuntos
* codigoOrigen Referencia
* codigoOrigenInterno Referencia interna
* cifrada Cifrado del cuerpo  del mensaje, Valores posibles:  true/false o 1/0
* referenciaAportacion Nº Referencia de la aportación
* unidadAdministrativa Descripción unidad Administrativa
* servicio Descripción servicio
* asunto Descripción asunto
* cuerpo Cuerpo de la notificacion
* codigoAbonado DUE del interesado
* lista de {NoteAdjuntos} listaAdjuntos
*/
public class NoteNotificacionRemesaListaAdjuntos
```

NoteAdjuntos.java

```
/**
* Objeto con los datos necesarios de un adjunto
* nombre
* fichero inputStreamDato {InputStream}
*/
public class NoteAdjuntos {
```

4.4. Método enviarRemesaAdjuntoUnico

NoteService.java

```
        /**
     * Método que envía una remesa de notificaciones con un único adjunto
     *
     * @param datosEntrada de tipo {@link NoteEnviarRemesaAdjuntoUnicoIN}
     * @throws ServiceException
     */
    public void enviarRemesaAdjuntoUnico(NoteEnviarRemesaAdjuntoUnicoIN
datosEntrada)
            throws ServiceException;
```

NoteEnviarRemesaAdjuntoUnicoIN.java

```
/**
* Objeto con los datos de entrada para el método enviarRemesaAdjuntoUnico
* {@link
atlas.eadm.services.NoteService#enviarRemesaAdjuntoUnico(NoteEnviarRemesaAdjuntoUnicoIN)}
* codigoAplicacion
* lista de {NoteNotificacionRemesaAdjuntoUnico} listaNotificaciones
*/
public class NoteEnviarRemesaAdjuntoUnicoIN {
```

NoteNotificacionRemesaAdjuntoUnico.java

```
/**
* Objeto con los datos necesarios para una notificación con un único adjunto
* codigoOrigen Referencia
* codigoOrigenInterno Referencia interna
* cifrada Cifrado del cuerpo  del mensaje, Valores posibles:  true/false o 1/0
* referenciaAportacion Nº Referencia de la aportación
* unidadAdministrativa Descripción unidad Administrativa
* servicio Descripción servicio
* asunto Descripción asunto
* cuerpo Cuerpo de la notificacion
* codigoAbonado DUE del interesado
* adjunto del tipo {NoteAdjuntos}
*/
public class NoteNotificacionRemesaAdjuntoUnico
```

NoteAdjuntos.java

```
/**
* Objeto con los datos necesarios de un adjunto
* nombre
* fichero inputStreamDato {InputStream}
*/
public class NoteAdjuntos {
```

4.5. Método obtenerNotificacionesPendientes

NoteService.java

```
    /**
     * Método que devuelve las notificaciones generadas por una aplicación
     *
     * @param datosEntrada String de código aplicación
     * @return {NoteObtenerNotificacionesPendientesOut}
     * @throws ServiceException
     */
    public NoteObtenerNotificacionesPendientesOut obtenerNotificacionesPendientes(
            String codigoAplicacion) throws ServiceException;
```

Objeto de salida
Los métodos que devuelven un objeto como salida, heredan de NoteRespuestaBase

NoteRespuestaBase.java

```
/**
 * Clase Base de la que heredarán todas las clases OUT de los
 * métodos del servicio
 *
 */
public class NoteRespuestaBase {
```

NoteObtenerNotificacionesPendientesOut.java

```
    /** Respuesta del método obtenerNotificacionesPendientes
 * {atlas.eadm.services.NoteService#obtenerNotificacionesPendientes}
 * Hereda de {NoteRespuestaBase}
 * Lista de NoteNotificacionPendiente
 *
 */

public class NoteObtenerNotificacionesPendientesOut extends NoteRespuestaBase {
```

NoteNotificacionPendiente.java

```
/**
* Objeto con los datos necesarios para una notificación pendiente
* identificador Identificador de la notificación
* codigoOrigen Referencia
* codigoOrigenInterno Referencia interna
* codigoAbonado Código del abonado
* fechaPuestaDisposicion Fecha de puesta a disposición
* firmaPuestaDisposicion Fecha de firma de puesta a disposición
* fechaLectura Fecha de la lectura
* firmaLectura Fecha de la firma de la lectura
* fechaRechazo Fecha del rechazo
* firmaRechazo Fecha de la firma del rechazo
* fechaError Fecha del error
* firmaError Fecha de la firma del error
*/
public class NoteNotificacionPendiente
```

4.6. Método obtenerNotificacionesPendientesConAsunto

NoteService.java

```
      /**
     * Método que devuelve las notificaciones generadas por una aplicación
     * devolviendo además el asunto
     *
     * @param datosEntrada String de código aplicación
     * @return {NoteObtenerNotificacionesPendientesConAsuntoOut}
     * @throws ServiceException
     */
    public NoteObtenerNotificacionesPendientesConAsuntoOut
obtenerNotificacionesPendientesConAsunto(String codigoAplicacion)
            throws ServiceException;
```

Objeto de salida

NoteObtenerNotificacionesPendientesConAsuntoOut.java

```
/** Respuesta del método obtenerNotificacionesPendientesConAsunto
 * {atlas.eadm.services.NoteService#obtenerNotificacionesPendientesConAsunto}
 * Hereda de {NoteRespuestaBase}
 * Lista de {NoteNotificacionPendienteConAsunto}
 *
 */

public class NoteObtenerNotificacionesPendientesConAsuntoOut extends
NoteRespuestaBase {
```

NoteNotificacionPendienteConAsunto.java

```
/**
* Objeto con los datos necesarios para una notificación pendiente con asunto
* identificador Identificador de la notificación
* codigoOrigen Referencia
* codigoOrigenInterno Referencia interna
* codigoAbonado Código del abonado
* fechaPuestaDisposicion Fecha de puesta a disposición
* firmaPuestaDisposicion Fecha de firma de puesta a disposición
* fechaLectura Fecha de la lectura
* firmaLectura Fecha de la firma de la lectura
* fechaRechazo Fecha del rechazo
* firmaRechazo Fecha de la firma del rechazo
* fechaError Fecha del error
* firmaError Fecha de la firma del error
* asunto Asunto de la notificación
*/
public class NoteNotificacionPendienteConAsunto
```

4.7. Método eliminarNotificacionesPendientes

NoteService.java

```
    /**
     * Elimina las notificaciones pendientes de la cola de mensajes generadas
     * por una aplicación.
     *
     * @param datosEntrada de tipo
     *            {@link NoteEliminarNotificacionesPendientesIN}
     * @throws ServiceException
     */
    public void
eliminarNotificacionesPendientes(NoteEliminarNotificacionesPendientesIN
datosEntrada)
            throws ServiceException;
```

Parámetro de entrada

NoteEliminarNotificacionesPendientesIN.java

```
/**
 * Objeto con los datos de entrada para el método
 * eliminarNotificacionesPendientes de
 * {@link
atlas.eadm.services.NoteService#eliminarNotificacionesPendientes(NoteEliminarNotificaciones
 * codigoAplicacion
 * lista de String listaIdentificadoresNotificaciones
 */
public class NoteEliminarNotificacionesPendientesIN {
```

4.8. Método obtenerDUE

NoteService.java

```
    /**
     * Devuelve la Dirección Electrónica Única asociada a los datos de un
     * interesado.
     *
     * @param codigoApp Código aplicación
     * @param apellidos Apellidos del abonado
     * @param nifCif N° de Identificación del abonado
     * @return String con resultado
     * @throws ServiceException
     */
    public String obtenerDUE(String codigoApp, String apellidos, String nifCif)
            throws ServiceException;
```

4.9. Método existeAbonado

NoteService.java

```
        /**
     * Comprueba si el interesado está dado de alta en el servicio de
     * notificaciones telemáticas.
     *
     * @param codigoApp Código aplicación
     * @param apellidos Apellidos del abonado
     * @param nifCif N° de Identificación del abonado
     * @return int con resultado
     * @throws ServiceException
     */
    public int existeAbonado(String codigoApp, String apellidos, String nifCif)
            throws ServiceException;
```

5. ENLACES RELACIONADOS

Producto	URL
Documentación EREG	http://intranet.madrid.org/arquitecturasw/admelectronica/EREG